Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

УТВЕРЖДАЮ
Заведующий кафедрой
_____ / В.В. Шайдуров

«17» июня 2016 г.

БАКАЛАВРСКАЯ РАБОТА

Направление 02.03.01Математика и компьютерные науки

АВТОМАТИЗАЦИЯ ПРОЦЕССА УПРАВЛЕНИЯ РАЗРАБОТКОЙ ПРИЛОЖЕНИЙ ПО ТЕХНОЛОГИИ КАНБАН

Научный руководитель
кандидат педагогических наук,
старший преподаватель _____ /Н.М. Андреева
17.06.2016

Выпускник _____ / Н.Н. Бурцев
17.06.2016

Красноярск 2016

РЕФЕРАТ

Выпускная квалификационная работа по теме «Автоматизация процесса управления разработкой приложений по технологии КАНБАН» содержит 26 страниц, 8 иллюстраций, 3 приложения и 18 использованных источников.

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ТЕХНОЛОГИЯ КАНБАН, КЛИЕНТ-СЕРВЕРНАЯ РАБОТА ПРИЛОЖЕНИЙ

Объект исследования – процесс разработки программного обеспечения. Предмет исследования – технология разработки программного обеспечения КАНБАН. Цель работы - разработка информационной клиент-серверной системы, которая реализует основные принципы технологии разработки программного обеспечения КАНБАН.

Задачи работы: создать информационную базу данных для хранения элементов технологии управления КАНБАН, организовать клиент-серверную работу информационной системы и отображение сущностей базы данных на клиенте.

Разработанные программные средства могут быть использованы в компаниях, занятых в области IT-инженерии.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Рынок разработки программного обеспечения появился совсем недавно – 50-60 лет назад. Стремительный рост информационных технологий подтолкнул многие компании к их использованию для увеличения доходности. Высокая конкуренция способствовала развитию идеи эффективного управления IT-проектами. В связи с этим сегодня все больше внимания уделяют различным технологиям разработки программного обеспечения (ПО), направленным на организацию и снижение издержек производственного процесса.

Динамичность изменений экономических условий работы предприятия, изменений его структуры требует мобильного подхода к разработке программного обеспечения. Для синхронизации команды разработчиков, территориально разобщенных, необходимо наличие централизованной информационной системы управления разработкой. Нелинейное взаимодействие разработчиков и пользователей обуславливает необходимость визуализации процесса разработки программного обеспечения.

Объект исследования – процесс разработки программного обеспечения.

Предмет исследования – система управления разработкой программного обеспечения КАНБАН.

Цель работы - разработка информационной клиент-серверной системы, которая реализует основные принципы технологии разработки программного обеспечения КАНБАН.

Задачи работы:

– создать информационную базу данных для хранения элементов технологии управления КАНБАН,

– организовать клиент-серверную работу информационной системы, отображение сущностей базы данных на клиенте.

1 Технологии разработки программного обеспечения

1.1 Каскадная, спиральная и итерационная технологии разработки программного обеспечения

Динамичность изменений экономических условий работы предприятия, изменений его структуры требует мобильного и технологичного подхода к разработке программного обеспечения.

Основные технологии разработки программного обеспечения: каскадная модель, спиральная модель, итерационная модель.

Каскадная модель процесса разработки программного обеспечения также часто называемой моделью “Водопада”, описанная У. У. Ройсом в 70-х годах прошлого столетия. Особенностью данного подхода является то, что переход на следующий этап возможен только в том случае, если предыдущий полностью завершен. Цикл разработки программного обеспечения включает в себя последовательное прохождение ряда этапов [1, с. 23]. Основные этапы [1, с. 47] процесса разработки программного обеспечения каскадной модели (Рисунок 1):

Формирование и анализ требований заключается в сборе требований к продукту.

Проектирование описывает внутреннюю структуру проекта.

Реализация – это программирование. Результатом является приложение готовое к тестированию.

Тестирование – проверка приложение на прохождение определенных качественных стандартов.

Ввод в действие – передача/установка приложения заказчику.

Сопровождение – исправление ошибок, которые не были обнаружены на этапе тестирования в виде создания патчей и передачи заказчику.

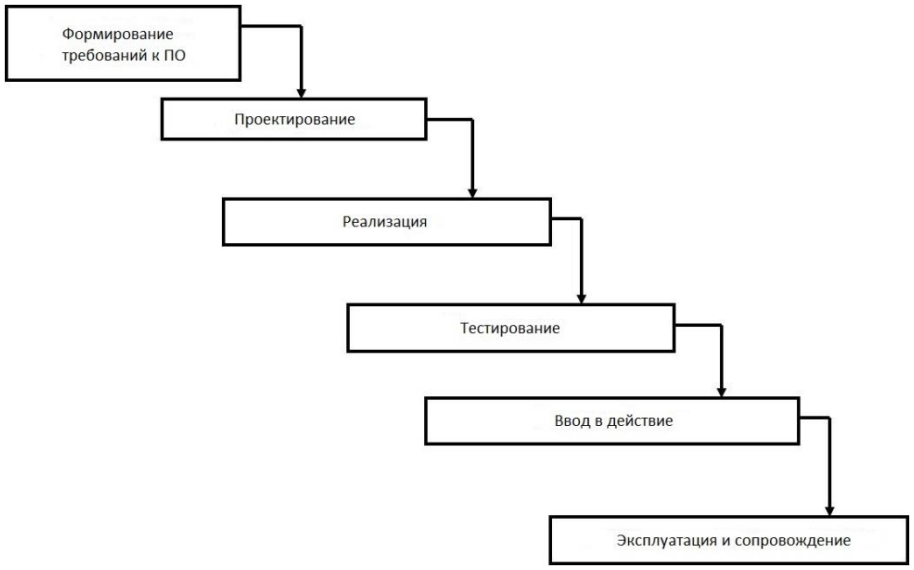

Рисунок 1 – Каскадная модель разработки программного обеспечения

Начинается с определения требований. Далее следует подробное описание того, что должно быть создано, так называемый анализ требований. Следующий этап - проектирование системы. Сюда входит выбор инструментов разработки, закладывание архитектуры будущего приложения. И заканчивается все тестированием и отладкой с последующим сопровождением [2, с. 75].

Сформулированные У.У. Ройсом основные этапы разработки программного обеспечения до сих пор остаются неизменными. Основным недостатком каскадной модели разработки является невозможность быстро реагировать на изменение факторов, влияющих на производственный процесс [3, с. 176]. Это может быть, как внешние факторы, например, экономическая ситуация в стране, так и внутренние – например, уход ведущего сотрудника из компании.

Особенностью *итеративной (эволюционной)* (Рисунок 2) модели разработки является непрерывающийся процесс анализа полученного опыта для оптимизации процесса разработки. Это достигается за счет разбития проекта на итерации, которые в приближении будут представлять микро каскадный цикл [4, с. 32]. Этапы итеративной модели аналогичны тем, которые используются в каскадной модели.

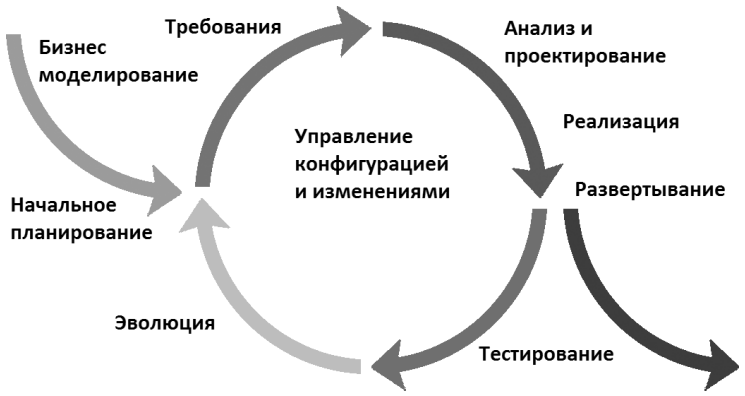

Рисунок 2 - Итерационная модель разработки программного обеспечения

Хотя каждая отдельная итерация может казаться последовательной, но реальная очередность этапов может быть нелинейной.

При этом требования могут меняться в ходе разработки [5, с. 133]. Итерации – реализация одного или нескольких функциональных требований. Результатом выполнения очередной итерации является прототип продукта с частичной функциональностью. Под *прототипом* понимается действующий программный компонент, реализующий отдельные функции и внешние интерфейсы разрабатываемого программного обеспечения. Особенности итерационной модели:

- стадии повторяются неоднократно,
- сначала для плохо сформулированных требований выполняется весь цикл работ по созданию работающего *прототипа*, затем уточняются требования; цикл работ повторяется,
- на выходе – продукт, отвечающий потребностям пользователей.

К недостаткам итерационной технологии можно отнести: слабую структурированность системы, «непрозрачность» проекта.

В 1988 году Барри Боэмом была предложена спиральная модель процесса разработки программного обеспечения, которая сочетает этапность и итеративность процесса разработки (Рисунок 3). Преимущество данного подхода заключается в снижении рисков на ранних этапах разработки, влияющих на успех проекта, путем уточнения требований и демонстрации проделанной работы заказчику после каждой итерации.

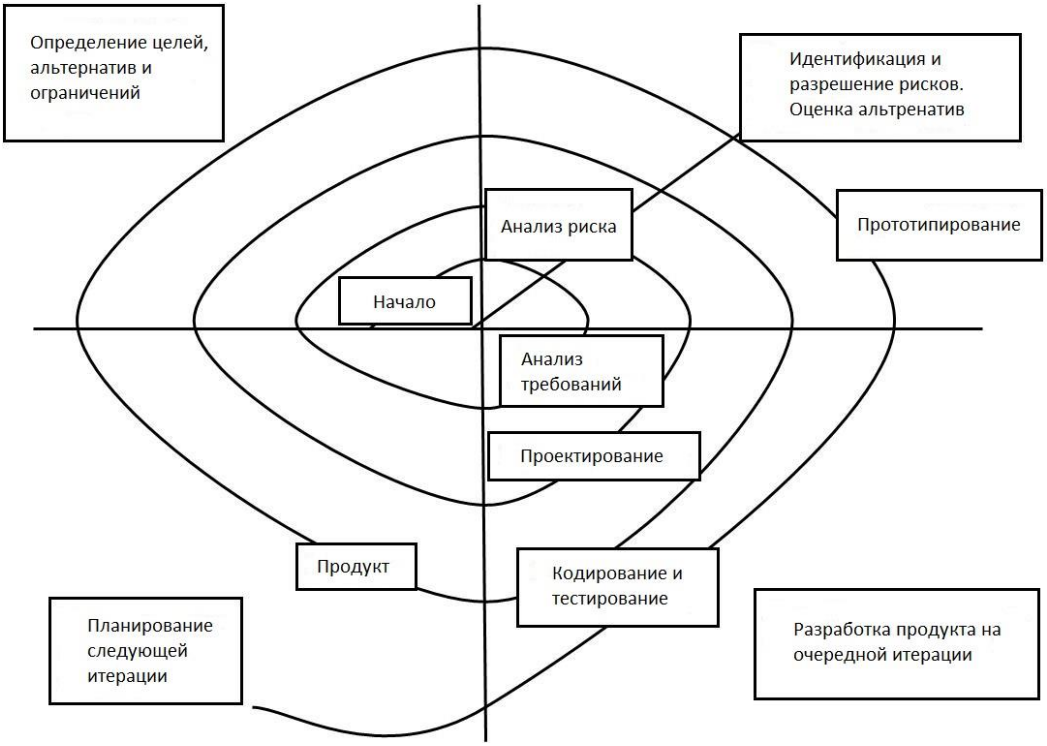

Рисунок 3 - Спиральная модель разработки программного обеспечения

Определение целей, альтернатив, ограничений, или фаза планирования. С этой стадии начинается работа над проектом. Формулируются цели проекта, основные требования, возможный дизайн и т.д. На последующих спиралях требования формируются с учетом корректировок от заказчика.

Анализ, определение и разрешение рисков является одной из самых значимых стадий разработки. В данном контексте, риски — это возможные события и состояния проекта, препятствующие достижению командой разработчиков поставленных целей [6]. Основной задачей для команды разработчиков является выявление всех возможных рисков и присвоение им определенного уровня приоритета. Продумываются стратегии преодоления этих рисков. В качестве результата работы на этом этапе создается *прототип*.

Фаза разработки. В этот этап входит разработка и последующее тестирование продукта. Во время первой итерации, когда общие требования еще не так четко сформулированы, разрабатывается концепция будущего продукта, которая необходима для получения отзыва заказчика. На следующих

витках в концепцию могут вносить коррективо по результатам получения отзыва от заказчика.

Планирование следующей фазы. На этом этапе вся полученная информация используется для планирования дальнейших этапов разработки.

В современных условиях на основе итеративной модели появились так называемые гибкие подходы к разработке программного обеспечения (Agile). Популярность данной технологии обусловлена тем, что позволяет быстрее адаптироваться перед изменяющимися требованиями, условиями работы. Короткие итерации позволяют быстрее вносить коррективы в процесс разработки. План проекта заменяется буфером задач, из которого вытягиваются задачи в начале каждой итерации, в зависимости от приоритетов на конкретном этапе развития проекта.

Развитие концепций гибкой разработки привело к различным самостоятельным моделям со своими особенностями: Scrum, Lean и т.д.

Scrum – технология для небольшим команд [7]. Проект делится на короткие итерации (спринты). В рамах каждого спринта команда берет на себя обязательство реализовать фиксированное количество нового функционала. Новые задачи в течение итерации откладываются на следующие спринты. Соблюдается строгий план выпуска релизов. Проведение каждый день коротких совещаний, на которых решаются возникшие проблемы.

Бережливое производство (Lean) - технология управления производственным процессом, основанная на постоянном стремлении к устранению всех видов издержек [8]. Основные идеи:

- определить ценность продукта,
- определить этапы создания ценности для этого продукта,
- обеспечить непрерывность процесса создания,
- позволить потребителю вытягивать продукт,
- стремиться к развитию.

Идеи бережливого производства (Lean) и принципы корпоративной работы нашли свое отражение в методе управления разработкой программного обеспечения КАНБАН.

1.2 Разработка IT-приложений по технологии КАНБАН

Данная технология была впервые использована в Японии в 1960-х годах автомобилестроительной компанией Toyota. Это система производства, нацеленная на устранение потерь, а также выявление более эффективных способов создания продукта [9, с. 44]. Основные принципы:

- конвейерный способ производства,
- ограничение количества незавершенной работы,
- концепция вытягивания задач,
- визуализация процесса разработки,
- непрекращающаяся оптимизация производственного процесса.

Для начала надо понять, почему данная концепция применима в контексте информационных технологий. Выше мы рассматривали каскадную модель производства программного обеспечения. Не трудно увидеть, что данная модель представляет собой конвейер, но вместо станков и машин выступают различные специалисты, через которых проходят не изделия, а задачи. Например, за подготовку и анализ требований будут отвечать руководство и аналитики. За проектирование и кодирование системы – разработчики. Тестирование – соответственно, тестировщики.

Основная идея КАНБАНА – количество незавершенной работы должно быть ограничено [10]. Следующий шаг производственного процесса не сможет начаться, пока не завершена текущая работа. Поэтому в случае проблем инициализируется «остановка конвейера», которая позволяет добиться снижения производственных издержек. Ведь если отправить некачественно выполненную задачу на следующий шаг, это повлечет за собой потерю времени

и ресурсов. Таким образом, происходит стимуляция для коллективного решения задач, а, следовательно, и сплочение коллектива.

Для визуализации может быть использована обычная доска в офисе, по которой будут двигаться стикеры-задачи. Но специфика разработки программного обеспечения состоит в том, что команда территориально разобщена. Для синхронизации всех специалистов необходимо наличие централизованной информационной системы управления разработкой.

Основные элементы технологии КАНБАН для разработки IT-приложений:

- фиксированная стадия Backlog, из которой вытягиваются задачи аналитиками, заказчиком, или человеком, который отвечает за развитие продукта,
- отобранные задачи – задачи, которые необходимо выполнить,
- распределенные задачи – находящиеся в разработке,
- реализованные задачи – готовые к передаче в отдел поддержки,
- установка – передача функционала заказчику,
- завершенные задачи – фиксированная стадия для определения, что задача закончила жизненный цикл.

Все элементы отражаются на доске КАНБАН. Технология КАНБАН дает возможность учитывать динамичность изменений экономических условий работы предприятия и его структуры, обеспечивает мотивацию и согласованность команды разработки программного обеспечения. Понятная визуализация процесса разработки программного обеспечения обеспечивает нелинейное взаимодействие разработчиков и пользователей.

2 Реализация процесса разработки по технологии КАНБАН на основе клиент-серверного приложения

2.1 Структура информационной базы для хранения элементов технологии КАНБАН

Основные этапы создания приложения: проектирование базы данных и описание основных сущностей, организация клиент-серверной работы информационной системы, отображение сущностей базы данных на клиенте.

Проектирование базы данных и описание основных сущностей заключается в отображении основных элементов КАНБАН: *доска* (Board), *стадия* (Stage), *строка* (Swim lane), *задача* (Task), *пользователь* (User), *роль* (Role) (Рисунок 4).

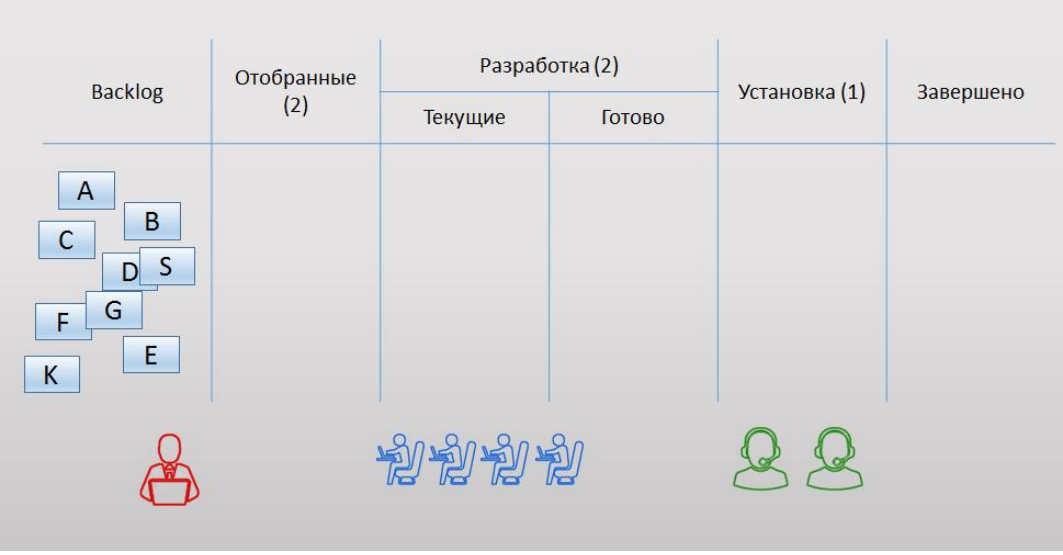

Рисунок 4 – Схема технологии КАНБАН для процесса разработки программного обеспечения

Доска – представляет собой КАНБАН в разрезе конкретного предприятия.

Стадия – производственный этап, через который должна пройти задача в процессе выполнения.

Задача – задание на реализацию конкретного функционала.

Строка – олицетворяет определенный релиз/продукт.

Пользователь – участник производственного процесса, который приписан к определенным стадиям.

Роль – используется для механизма ограничений в информационной системе.

В основе клиент-серверной работы разрабатываемой информационной системы лежит двухзвенная клиент-серверная архитектура (Рисунок 5).

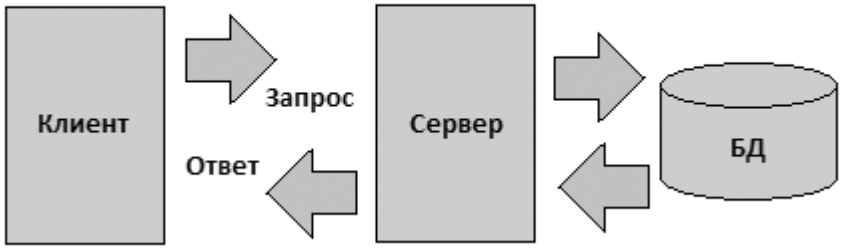

Рисунок 5 – Двухзвенная клиент-серверная архитектура [11]

Роль сервера заключается в обработке клиентских запросов и работе с базой данной (получение и запись информации обеспечивают SQL-запросы). Клиентами выступают компьютеры сотрудников компании.

Отображение сущностей базы данных на клиенте представлено на рисунке 6.

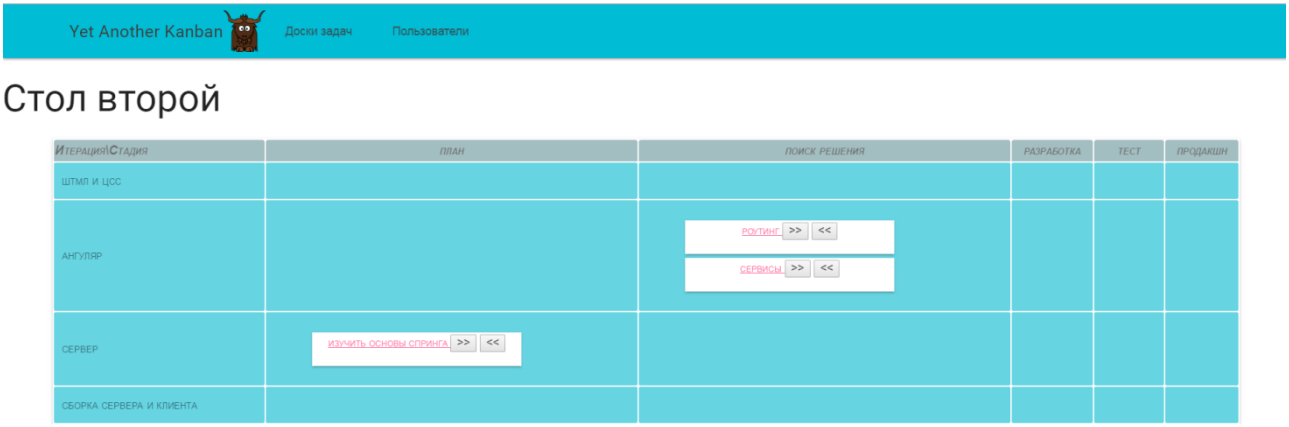

Рисунок 6 – Клиентский интерфейс информационной системы

2.2 Инструменты и средства для организации клиент-серверной работы информационной системы и отображения сущностей базы данных на клиенте

Основные инструменты и средства для организации клиент-серверной работы информационной системы и отображения сущностей базы данных на клиенте:

- к серверной части относятся: использование языка программирования Java [12], библиотека для java платформы Spring [13], база данных PostgreSQL [14], веб-сервер Tomcat [15];
- клиент-серверную работу обеспечивают: язык разметки HTML5, библиотека Angular, сценарный язык программирования JavaScript, CSS;
- кроме того, используются: система сборок Maven [16], система контроля версиями Git [17].

Основные функциональные блоки Spring’a: функционал управления транзакциями, функционал для доступа к базам данных, механизм настройки интерфейсов сервлетов (для расширения функциональных возможностей сервера).

Клиентом служит HTML-страница с использованием библиотеки Angular, которая позволяет описывать интерфейс приложения в декларативном стиле

(Приложение А). Для автоматической синхронизации модели и представления используется двусторонний биндинг.

2.3 Работоспособность информационной системы автоматизации процесса управления разработкой приложений по технологии КАНБАН

На текущий момент проведена предварительная настройка web-server'а Tomcat для возможности отладки приложения через среду разработки Eclipse (Рисунок 7).

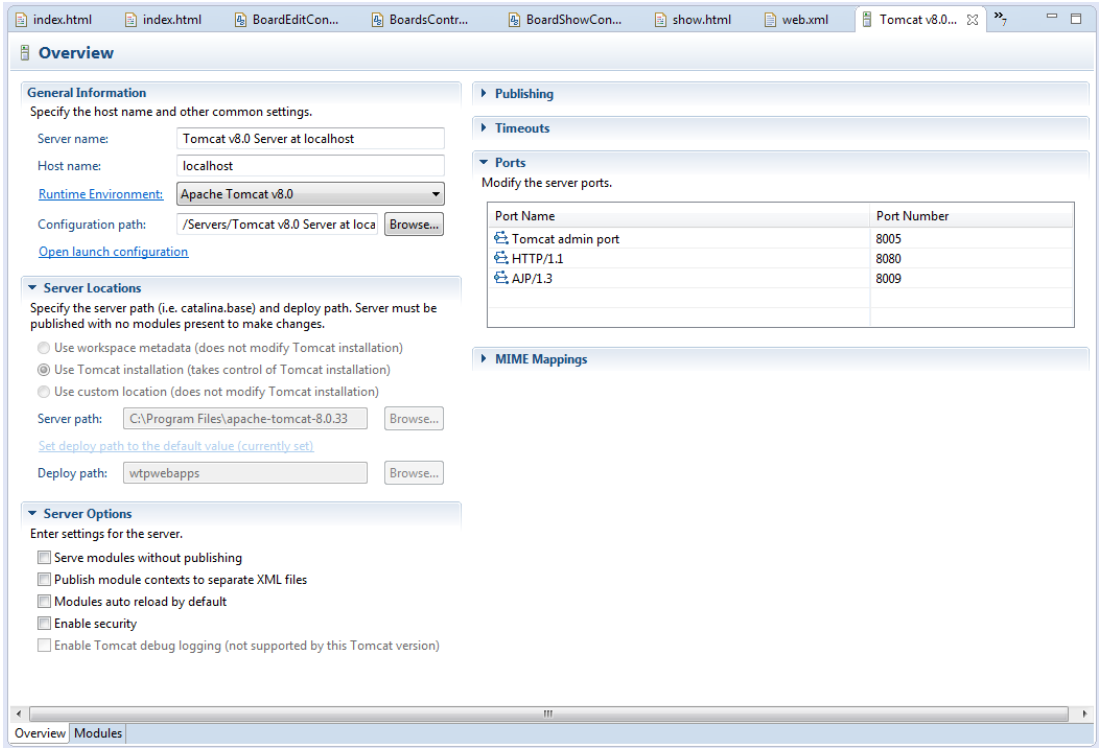

Рисунок 7 – Настройки сервера Tomcat для возможности отладки через среду разработки Eclipse

Произведена настройка базы данных PostgreSQL через приложение pgAdmin3 (Рисунок 8).

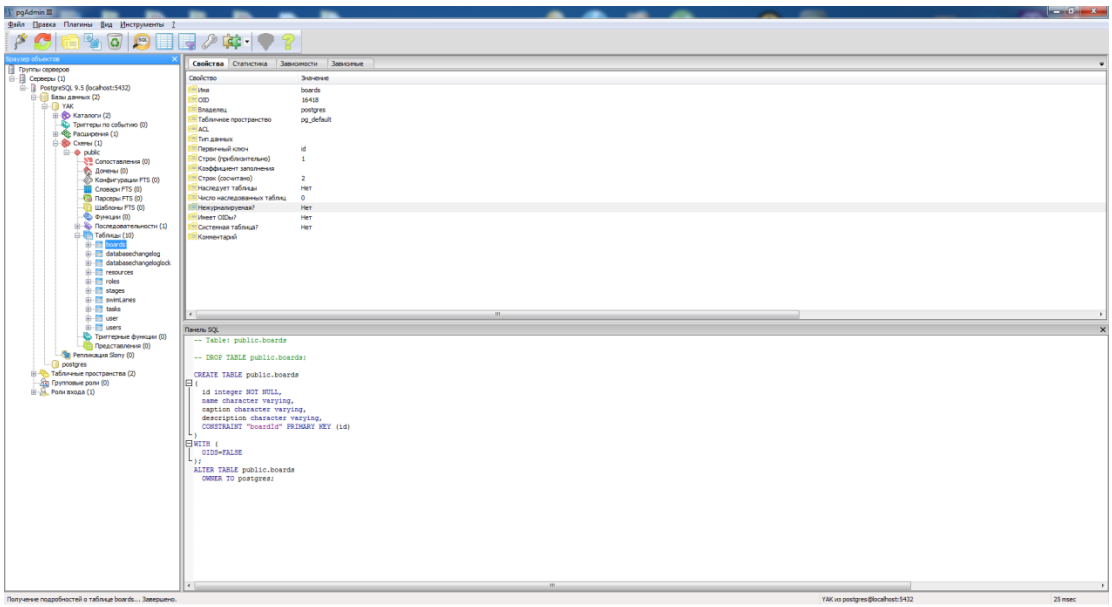

Рисунок 8 – Сущности в базе данных

Произведение формирование конфигурационного файла для сборки через систему Maven (Приложение Б). На сервере написаны базовые классы для получения данных из базы данных (Приложение В).

Предложенная информационная система позволяет автоматизировать производственный процесс разработки программного обеспечения по технологии КАНБАН. Основной целью системы является повышение качества исполнение процесса. Основные преимущества системы [18]:

- согласованность процесса разработки на каждом этапе,
- создание общей среды для работы организации,
- повышение качества работы сотрудников.

Разработанная система найдет применение среди предприятий, которые разрабатывают IT-приложения.

ЗАКЛЮЧЕНИЕ

Создана информационная система, автоматизирующая процесс разработки IT-приложений по технологии КАНБАН. Она обеспечивает:

- автоматизацию процесса управления разработкой программного обеспечения;
- мотивацию и согласованность команды разработки программного обеспечения;
- хранение количественных показателей успешности процесса разработки, формирование качественной оценки работы команды в целом и каждого в отдельности.

Разработанные программные средства могут быть использованы в компаниях, занятых в области IT-инженерии.

Результаты исследований, проведенных в рамках бакалаврской работы, были представлены в докладе «Автоматизация процесса управления разработкой программного обеспечения по технологии КАНБАН» на Международной научно-технической конференции студентов, аспирантов и молодых учёных «Проспект Свободный — 2016» и опубликованы в материалах этой конференции.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Брауде, Э. Д. Технология разработки программного обеспечения / Э. Дж. Брауде ; [пер. с англ. Е. Бочкаревой, Д. Солнышкова]. – Санкт-Петербург : Питер, 2004. – 654 с.

2. Гантер, Ричард. Методы управления проектированием программного обеспечения / Р. Гантер ; пер. с англ. Е. К. Масловского. – Москва : Мир, 1981. – 388 с.

3. Константин, Л. Человеческий фактор в программировании / Ларри Л. Константин; пер. Ю. Асотова. – Санкт-Петербург ; Москва : Символ-Плюс, 2004. – 382 с.

4. Разработка программных проектов на основе Rational Unified Process (RUP) / Гари Полис [и др ; предисл. Ф. Круктена] ; пер. с англ. под ред. А. П. Караваева. – Москва : Бином, 2011. – 255 с.

5. Фокс, Д. Программное обеспечение и его разработка / Д. Фокс ; пер. с англ. Л. Е. Карпова ; под ред. Д. Б. Подшивалова. – Москва : Мир, 1985. – 368 с.

6. Гурендо, Д. Жизненный цикл разработки ПО (SDLC). Спиральная модель [Электронный ресурс] : URL: http://xbsoftware.ru/blog/zhiznennyj-tsykl-razrabotki-spiral/ (дата обращения: 12.04.2016).

7. Методологии разработки программного обеспечения [Электронный ресурс] : URL: https://habrahabr.ru/sandbox/43802/ (дата обращения: 02.05.2016).

8. Бережливое производство // Википедия. [Электронный ресурс] : URL: https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%80%D0%B5%D0%B6%D0%BB%D0%B8%D0%B2%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE (дата обращения: 02.05.2016).

9. Канбан и "точно вовремя" на Toyota : менеджмент начинается на рабочем месте / пер. с англ. [Е. Пестерева] ; науч. ред. Ю. П. Адлер]. – Москва : Альпина Бизнес Букс, 2008. – 217 с.

10. Книберг, Х. Kanban и Scrum: выжимаем максимум / Хенрик Книберг, Маттиас Скарин ; предисл. Мэри Поппендик, Дэвида Андерсона ; пер. с англ. выполнен сообществом Agile Ukraine: Алексей Солнцев и др. [Электронный ресурс] : URL: http://scrum.org.ua/wp-content/uploads/ScrumAndKanbanRuFinal.pdf (дата обращения 15.04.2016).

11. Анатольев, А. Компоненты сетевого приложения. Клиент-серверное взаимодействие и роли серверов [Электронный ресурс] : URL: http://www.4stud.info/networking/lecture5.html (дата обращения 15.04.2016).

12. Java // Википедия. [Электронный ресурс] : URL: https://ru.wikipedia.org/wiki/Java (дата обращения: 02.05.2016).

13. Spring Framework [Электронный ресурс] : URL: http://projects.spring.io/spring-framework/ (дата обращения 15.04.2016).

14. DataBase PostgreSQL [Электронный ресурс] : URL: https://www.postgresql.org/ (дата обращения 14.04.2016).

15. Apache Tomcat Web-server [Электронный ресурс] : URL: http://tomcat.apache.org/ (дата обращения 15.04.2016).

16. Maven Home [Электронный ресурс] : URL: https://maven.apache.org/ (дата обращения 15.04.2016).

17. Git Home [Электронный ресурс] : URL: https://git-scm.com/ (дата обращения 15.04.2016).

18. Автоматизация процессов [Электронный ресурс] : URL: http://www.kpms.ru/Automatization/Process_automation.htm (дата обращения 15.04.2016).

ПРИЛОЖЕНИЕ А

Листинг просмотра КАНБАНа

```
<div >

<h2>{{boards[id].caption}}</h2>

<table class="mdl-shadow--2dp" >
          <thead>
               <th>Итерация\Стадия</th>
               <th  ng-repeat="stage in stages">{{stage.caption}}
               </th>
          </thead>
          <tr ng-repeat="milestone in milestones" style="height: auto;">
               <td class="mdl-data-table__cell--non-numeric">
                    {{milestone.caption}}
                    <!--input type="button" value="+"></input-->
               </td>
               <td ng-repeat="stage in stages" class="mdl-data-
table__cell--non-numeric">
                    <!--div style="vertical-align: top; display: block;
border-style: solid; text-align: right;" >+ task</div-->
                    <ul>

                         <li draggable ng-repeat="task in
getTasks(stage, milestone)"
                         class="taskitem mdl-shadow--2dp">
                                   <a ng-
href="#/boards/{{id}}/tasks/{{task.name}}" > {{task.name}} </a>

                                   <input type="button" ng-
click="setNextStage(task);" value=">>"></input>
                                   <input type="button" ng-
click="setPrevStage(task);" value="<<"></input>
                                   <!--input type="button" ng-
click="changeStage(task);" value="^^"></input>
                                   <input type="button" ng-
click="changeStage(task);" value="VV"></input-->
                         </li>
                    </ul>
               </td>
          </tr>
     </table>

</div>


<ul class="mdl-list" >
```

```
      <li ng-repeat="board in boards" class="mdl-list__item">

  <div>
    <div>
    <a ng-href="#/boards/{{board.id}}" > {{board.caption}} </a>
    </div>
    <div>
    <a ng-href="#/boards/edit/{{board.id}}" > Редактировать </a>
    </div>
  </div>

      </li>
</ul>
```

ПРИЛОЖЕНИЕ Б

Конфигурационный файл POM

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">

	<context-param>
		<param-name>contextConfigLocation</param-name>
		<param-value>
			/WEB-INF/spring/root-context.xml
		</param-value>
	</context-param>

	<listener>
		<listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
	</listener>
	<!-- <listener> -->
	<!-- <listener-
class>org.springframework.security.web.session.HttpSessionEventPublisher<
/listener-class> -->
	<!-- </listener> -->
	<servlet>
		<servlet-name>appServlet</servlet-name>
		<servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
		<init-param>
			<param-name>contextConfigLocation</param-name>
			<param-value>/WEB-INF/spring/servlet-context.xml</param-
value>
		</init-param>
		<load-on-startup>1</load-on-startup>
	</servlet>
	<servlet-mapping>
		<servlet-name>appServlet</servlet-name>
		<url-pattern>/</url-pattern>
	</servlet-mapping>

	<filter>
```

```
            <filter-name>charsetFilter</filter-name>
            <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
            <init-param>
                  <param-name>encoding</param-name>
                  <param-value>UTF-8</param-value>
            </init-param>
            <init-param>
                  <param-name>forceEncoding</param-name>
                  <param-value>true</param-value>
            </init-param>
      </filter>
      <filter-mapping>
            <filter-name>charsetFilter</filter-name>
            <url-pattern>/*</url-pattern>
      </filter-mapping>


      <resource-ref>
            <description>postgreSQL Datasource example</description>
            <res-ref-name>jdbc/mainDB</res-ref-name>
            <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
</resource-ref>
</web-app>
```

ПРИЛОЖЕНИЕ В

Базовый класс реализация доступа к данным

```
import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.ObjectNotFoundException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;

public abstract class BaseDAOImpl<Entity extends PersistentEntity>
implements BaseDAO<Entity> {
	private Class<Entity> type;

	@Autowired
	protected SessionFactory sessionFactory;

	public BaseDAOImpl(Class<Entity> type) {
		this.type = type;
	}

	protected Session getSession() {
		return sessionFactory.getCurrentSession();
	}

	protected Criteria createCriteria() {
		return getSession().createCriteria(type);
	}

	@Override
	public void save(Entity obj) {
		getSession().save(obj);
	}

	@Override
	public List<Entity> list() {
		return
ObjectsSupport.safeCastList(getSession().createCriteria(type).list());
	}

	@Override
	public Entity get(Long id) {
		Entity obj = find(id);
		if (obj == null) {
			throw new ObjectNotFoundException(id,
type.getSimpleName());
		}
		return obj;
```

```
	}

	@Override
	public void delete(Long id) {
		Object obj = getSession().load(type, id);
		if (obj == null) {
			throw new ObjectNotFoundException(id,
type.getSimpleName());
		}
		getSession().delete(obj);
	}

	@Override
	public void deleteAll() {
		getSession().createQuery("delete from " +
type.getSimpleName()).executeUpdate();
	}

	@Override
	public void saveOrUpdate(Entity obj) {
		getSession().saveOrUpdate(obj);
	}

	@Override
	public void update(Entity obj) {
		getSession().update(obj);
	}

	@Override
	public Entity find(Long id) {
		return type.cast(getSession().byId(type).load(id));
	}

	public Entity first(List<Entity> list)
	{
		if (list.size()>0)
		{
			return list.get(0);
		}
		else
		{
			return null;
		}
	}
}
```